Supplementary Data for "Efficient Assembly of Large Genomes using Single-Molecule Sequencing and Locality Sensitive Hashing"

Online Resources

Pre-compiled source code and datasets used for this publication:
http://www.cbcb.umd.edu/software/PBcR/MHAP

Celera Assembler source code and documentation (including PBcR correction pipeline):
http://wgs-assembler.sourceforge.net

PBcR correction pipeline documentation:
http://sourceforge.net/apps/mediawiki/wgs-assembler/index.php?title=PBcR

Table of Contents

Supplementary Note 1: Overlap analysis

In order to determine the sketch size and *k*-mer size required to effectively discriminate between true and false matches, we developed a simulator to test Jaccard similarity. Sequences were simulated from a given reference genome and Jaccard similarity was measured for true and random matches. Repetitive *k*-mers can optionally be ignored when calculating similarity. The simulator is included in the MHAP source code and can be executed as:

```
java edu.umd.marbl.mhap.main.KmerStatsSimulator 50000 <10 or 16> 10000 5000
0.1188 0.0183 0.0129 false <reference.fasta>
```

to simulate pairwise overlap error rates, or as:

```
java edu.umd.marbl.mhap.main.KmerStatsSimulator 50000 <10 or 16> 10000 5000
0.1188 0.0183 0.0129 true <reference.fasta>
```

to simulate mapping error rates. For Figure 2 in the main manuscript, GRCh37 was used to select random sequences with no filtering for repetitive *k*-mers.

To examine the performance for increasing sequence lengths, we simulated 620Mbp (0.2X) of sequences from 5Kbp to 100Kbp in length. The sequences can be simulated using the command:

```
java edu.umd.marbl.mhap.main.KmerStatsSimulator <# sequences to generate>
<sequence length> 0.10 0.02 0.01 <reference.fasta>
```

Finally, we ran the "direct" implementation with output disabled as:

```
java edu.umd.marbl.mhap.main.DirectAlignMain –k 16 --num-hashes 0 --num-min-
matches 100000 --threshold 0.03 --subsequence-size 100000 --min-store-length
2000 --num-threads 32 -s <simulated_sequences.fasta>
```

and our MinHash implementation as:

```
java edu.umd.marbl.mhap.main.FastAlignMain –k 16 --num-hashes <512/1256> --
num-min-matches 3 --threshold 0.04 --subsequence-size 100000 --min-store-
length 2000 --num-threads 32 -s <simulated_sequences.fasta> > <mhap.ovls>
```

As these were simulated sequences, the truth was already known and sensitivity was evaluated as described in the methods section of manuscript:

```
 java edu.umd.marbl.mhap.main.EstimateROC <truth.m4> <mhap.ovls>
<simulated_sequences.fasta> <min overlap> 10000 true false
```

The minimum overlap size was held constant at 20% of sequence length. Sensitivity of MHAP averaged 53% for sketch-size=512 and 80% for sketch size=1256.

Sequence Length	# Sequences	Overlap Length	Sensitivity (512)	Sensitivity (1256)	Specificity (both)
5000	124,073	1000	50%	77%	100%
10000	62,037	2000	52%	79%	100%
15000	41,358	3000	51%	76%	100%
20000	31,019	4000	52%	76%	100%
25000	24,815	5000	53%	79%	100%
30000	20,679	6000	54%	80%	100%
35000	17,725	7000	54%	80%	100%
40000	15,510	8000	55%	80%	100%
45000	13,786	9000	57%	84%	100%
50000	12,408	10000	53%	70%	100%
55000	11,280	11000	53%	77%	100%
60000	10,340	12000	53%	85%	100%
65000	9,545	13000	53%	84%	100%
70000	8,863	14000	53%	83%	100%
75000	8,272	15000	52%	83%	100%
80000	7,755	16000	53%	82%	100%
85000	7,299	17000	52%	81%	100%
90000	6,893	18000	53%	80%	100%
95000	6,531	19000	53%	80%	100%
100000	6,204	20000	53%	78%	100%

Table S1. MHAP sensitivity across varying sequence lengths. Sequences were simulated from the human genome varying in size from 5Kbp to 100Kbp (by 5Kbp) and sensitivity evaluated given a fixed overlap percentage (20%). Sensitivity is given for sketch sizes of 512 and 1256.

Performance Across Varied Parameters

The sensitivity and accuracy of overlaps was measured on *E. coli* using three available chemistry datasets (C2, P4-C2, and P5-C3) (Supplementary Table S2, Fig. S1). MHAP was run with varying *k*-mer size (10-16), # hashes (64-1512), and minimum # matches (2-4). BLASR was run with varying *k*-mer size (10-16) and # best matches (C, 1C, 5C, 10C, 100C), where C represents approximate genome coverage (100X in this case). The same analysis was repeated on a random subset of *A. thaliana*. A random subset of sequences adding up to 500Mbp was selected and parameters evaluated (Supplementary Table S2, Fig. S2).

Supplementary Note 2: Comparison of overlapping tools

The latest versions as of July 2014 of BWA-MEM (0.7.9a), RazerS (3.1.1), SNAP (1.0beta.10), and BLASR (SMRTanalysis 2.2.0) were downloaded. A simulated dataset of 543 sequences from lambda phage was used to evaluate both mapping to reference and overlap detection.

First, programs were run to map sequences to the reference to ensure they were compatible with noisy, long sequences:
BWA-MEM was run as:

```
bwa mem -t 16 -x pacbio lambda.fasta lambda.sim.varyLength.fasta > bwamem.sam
2> bwamem.out
```

BLASR was run as:

```
blasr -nproc 16 -maxLCPLength 16 -minMatch 12 -m 4 -bestn 1
lambda.sim.varyLength.fasta lambda.fasta > blasr.m4 2>blasr.err
```

RazerS was run as:

```
razers3 -i 85 -tc 16 -o ill.razers ../lambda.fasta
lambda.sim.varyLength.fasta
```

SNAP was run as:

```
snapxl single lambda_snap/ lambda.sim.varyLength.fastq -o snap.sam  -d 700 -t
16
```

Both BLASR and BWA-MEM performed acceptably. SNAP was unable to align a majority of the sequences to the reference genome due to a compile-time restriction of the edit distance to 63. After recompiling to increase this threshold, a total of 65% of sequences could be mapped to the reference. RazerS did not complete after 30 minutes and was excluded from further testing. Next, sequences were aligned to each other to identify pairwise overlaps:
BWA-MEM was run as:

```
bwa mem -t 16 -x pbread lambda.sim.varyLength.fasta
lambda.sim.varyLength.fasta > bwamem_self.out 2> bwamem_self.err
```

BLASR was run as:

```
blasr -nproc 16 -maxLCPLength 16 -minMatch 12 -m 4 -bestn 50
lambda.sim.varyLength.fasta lambda.sim.varyLength.fasta > blasr.m4
2>blasr.err
```

SNAP was run as:

```
snapxl single output lambda.sim.varyLength.fastq -o snap.sam  -d 2900 -t 16
```

BWA-MEM sensitivity was 64% and was excluded from further analysis due to low sensitivity on this dataset. SNAP only detected 221 overlaps out of a total of over 25K and was also excluded. Both BLASR and MHAP achieved over 90% sensitivity.

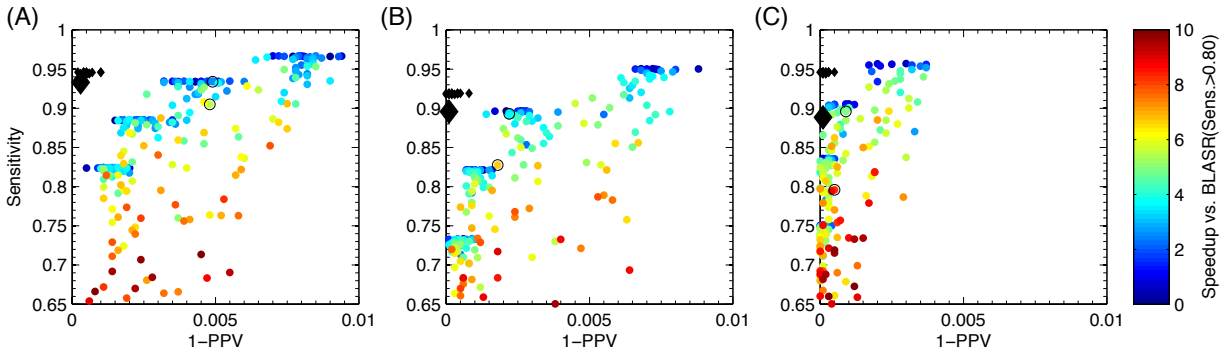

Figure S1: ROC plots for MHAP and BLASR with varying parameters for *E. coli* K12
The sequences were aligned to the reference genome to obtain the true sequence locations. Varied MHAP parameters were: *k*-mer size (12–16), sketch-size (256–1512), minimum matching min-mers (2–4), and minimum Jaccard threshold (0.02–0.04). Varied BLASR parameters were: *k*-mer size (10–16) and bestn (1C–100C, where *C* is the sequencing depth). Two reads whose mapping overlapped on the reference for more than 2,000bp were considered overlapping. Any reads whose mapping did not overlap, but were identified as overlapping by MHAP or BLASR were considered true when an overlap greater than 70% identity was confirmed by a Smith-Waterman alignment of the region (Methods). Therefore, repeat-induced matches were not considered false positives. Predicted overlaps were evaluated against this standard to compute positive predictive value (PPV) and sensitivity. BLASR data points are represented by black diamonds, with the largest indicating the defaults. Colored circles are MHAP data points. The circled points indicate the two MHAP defaults (fast or sensitive). Speedups for MHAP are given relative to the fastest BLASR run that achieved a minimum of 80% sensitivity. Results are given for three PacBio SMRT chemistries: (A) C2 chemistry, (B) P4C2 chemistry, and (C) P5C3 chemistry.

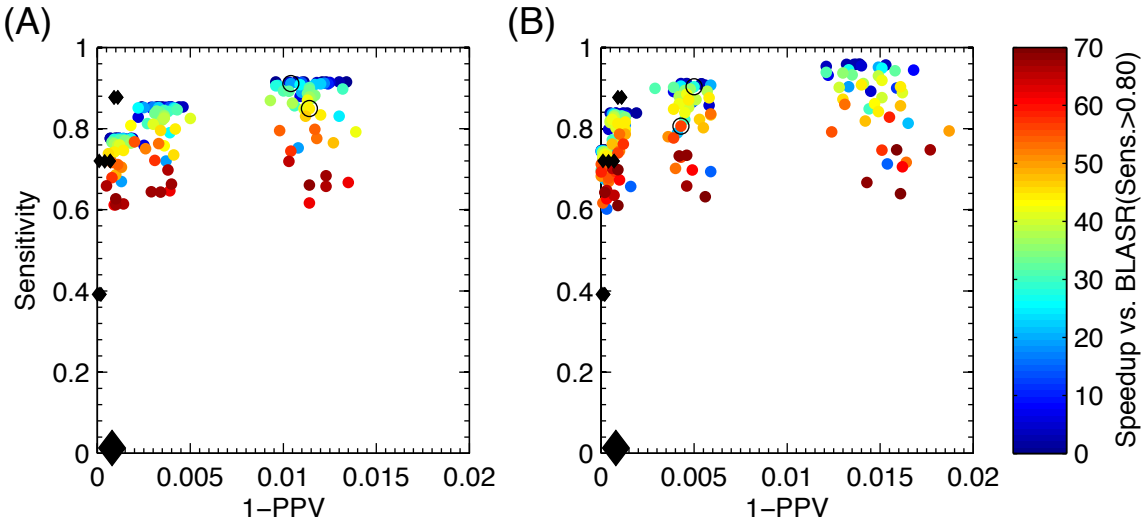

Figure S2: ROC plots for MHAP and BLASR with varying parameters for *A. thaliana*
Same plot as Figure S1 but for *A. thaliana* Ler-0 sequencing using (A) P4C2 chemistry and (B) P5C3 chemistry. For this dataset, only BLASR parameters (k=10–16, bestn=250C) achieved >80% sensitivity on this dataset.

Supplementary Note 3: Sequencing data

Genome	Download	Reference	Coverage	# Sequences	Chemistry	Mean (bp)	Max (bp)
Escherichia coli K12	SRX255228	NC_000913	86X	161,791	C2	2,472.91	14,632
Escherichia coli K12	SRX669475	NC_000913	94X	82,590	P4C2	5291.09	22,609
Escherichia coli K12	SRX533603	NC_000913	85X	47,910	P5C3	8282.76	28,647
Saccharomyces cerevisiae W303	SRX533604	NC_001133:NC_001148	117X	232,230	P4C2	6039.55	30,164
Arabidopsis thaliana Ler-0	SRX533608	TAIR10	110X	3,444,146	P4C2	4,142.16	41,753
Arabidopsis thaliana Ler-0	SRX533607	TAIR10	144X	2,021,431	P5C3	8,572.91	47,445
Drosophila melanogaster ISO1	SRX499318	Ref v5	121X	1,689,814	P5C3	9,317.04	44,766
Human CHM1htert	SRX533609	Rev v38	54X	22,565,609	P5C3	7,447.12	42,774

Table S3. Sequence data used to test PBcR-MHAP. The five datasets used for testing the PBcR-MHAP correction and assembly pipeline. The raw sequence lengths are reported before correction.

We tested the PBcR-MHAP algorithm on five genomes sequenced using PacBio SMRT sequencing. These datasets were generated and publically released by Pacific Biosciences and can be downloaded from https://github.com/PacificBiosciences/DevNet/wiki/Datasets or NCBI SRA. For *E. coli* and *A. thaliana*, only the P5C3 datasets were used to generate the assemblies reported.

Supplementary Note 4: Assembly validation

All genomes, except human, were aligned to their reference genomes and plotted using Nucmer (Figs. S3–S7). The commands to generate alignments were:

```
nucmer –mumref –l 100 –c 1000 –d 10 –banded –D 5 <ref.fasta> <asm.fasta>
delta-filter –i 95 –o 95 out.delta > out.best.delta
dnadiff –d out.best.delta
```

Human alignments were generated with:

```
nucmer –mumref –l 100 –c 1000 <ref.fasta> <asm.fasta>
delta-filter –i 95 –o 95 out.delta > out.best.delta
dnadiff –d out.best.delta
```

We used GAGE metrics to evaluate *E. coli* and *D. melanogaster* (two genomes where the same strain was used to construct the reference as was assembled) (Table S4). To identify contig counts for each reference chromosome the command:

```
cat out.1delta |awk '{if ($7 > 99 && $5 > 200) print $NF" "$(NF-1)}' |sort
|uniq |awk '{print $NF}' |sort  |uniq -c
```

Genome	Reference	CN50	# SNPs	# Indels <5bp	QV (SNP+Indel)	# Indels >5bp	Inversions	Relocations	Translocations
E. coli K12 pre-quiver	NC_000913	838,903	6	551	39.21	3	0	0	0
E. coli K12 post-quiver	NC_000913	838,907	3	8	56.25	3	0	0	0
E. coli K12 SPAdes	NC_000913	132,561	75	21	46.84	4	0	1	0
D. melanogaster pre-quiver	Ref v5	303,294	14,985	141,773	29.18	3,152	17	85	20
D. melanogaster post-quiver	Ref v5	645,364	3,141	8,338	40.53	357	17	63	7

Table S4. GAGE metrics for *E. coli* and *D. melanogaster*. The assemblies were aligned to the reference genome as above and GAGE assembly correctness metrics were calculated. Perceived errors due to circular chromosomes were manually excluded. The *D. melanogaster* ArmU and ArmUextra were excluded from validation as they contain unordered sequences which cannot be used for continuity validation.

To identify resolved gaps in *D. melanogaster,* the v5 reference was split at ten or more contiguous Ns and each break considered a gap, for a total of 124 gaps. Based on the Nucmer alignment generated above, a contig with at least 500bp leading up to and 500bp after the gap was considered to potentially close the gap. Finally, the size of the patch in the contig was compared to the size of the reference gap. The patch was considered to match the reference gap length if the reference gap was exactly 100bp (an artificial gap length) or within 25% of the patch size.

Human gaps were restricted to the annotated gap table downloaded from UCSC genome browser (https://genome.ucsc.edu/cgi-bin/hgTables?hgsid=384421183_2auF2x0v1KrkW2F051b7StoGPaOU) for Ref38 (a total of 819 gaps) and the same procedure followed. Further validation is required to determine if these patch sequences can be incorporated into the reference sequences.

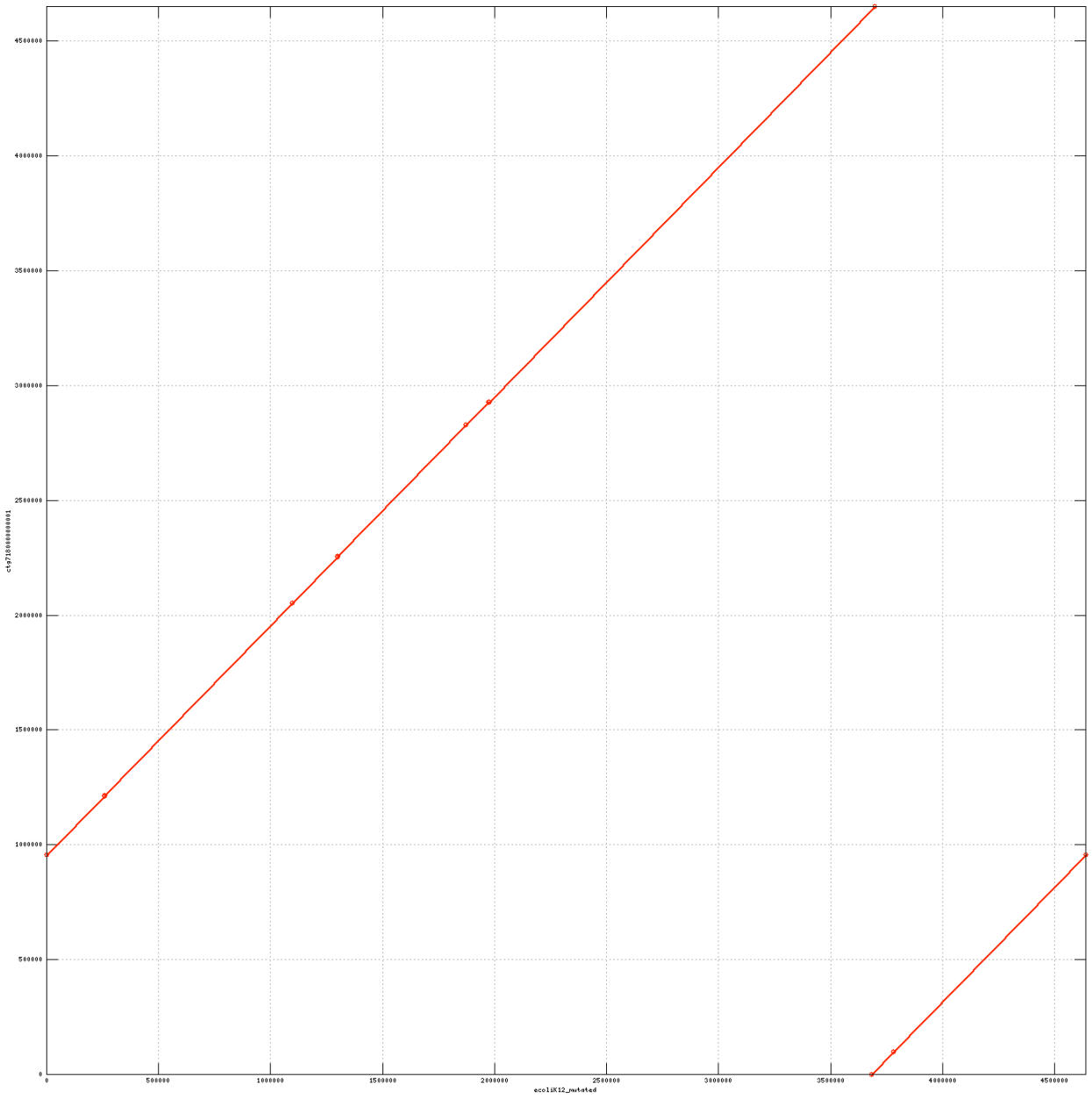

Figure S3. Mummerplot of the PBcR-MHAP assembly and *E. coli* reference
An alignment dotplot shows the relationship between the PBcR assembled contig of *E. coli* K12 (y-axis) and the *E. coli* K12 reference genome (x-axis). The single-contig assembly matches the reference over the entire length of the genome. The origin of the assembly is arbitrarily shifted due to the chromosome being circular, and does not represent an assembly error.

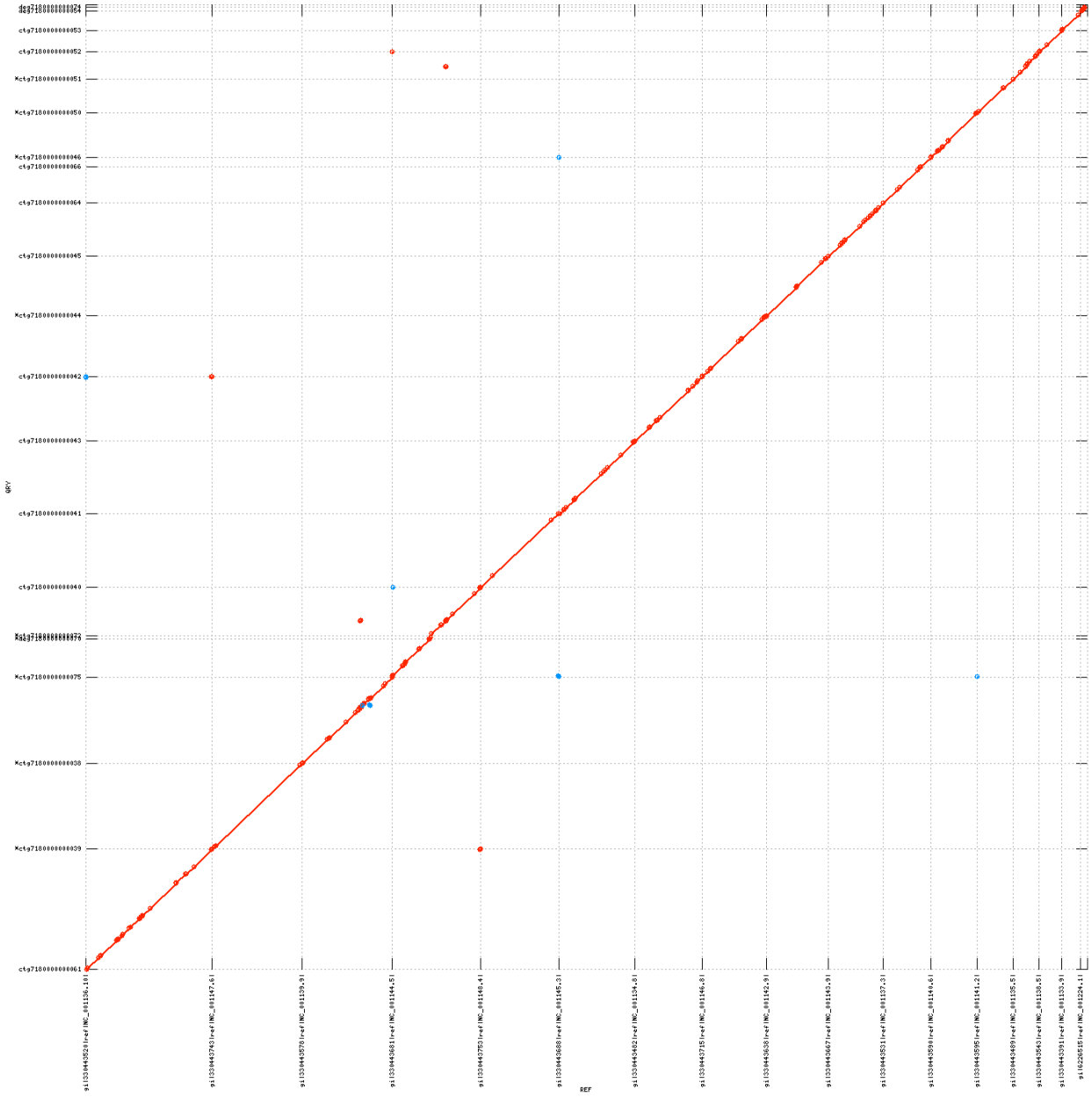

Figure S4. Mummerplot of the PBcR-MHAP assembly and *S. cerevisiae* reference
An alignment dotplot shows the relationship between the PBcR assembled contigs of *S. cerevisiae* W303 (y-axis) and the *S. cerevisiae* S228 reference genome (x-axis). Contig and chromosome boundaries are displayed as dotted lines (horizontal and vertical, respectively), and contigs are ordered and oriented to match the reference using mummerplot's –fat option. The majority of chromosomes are assembled into a single contig.

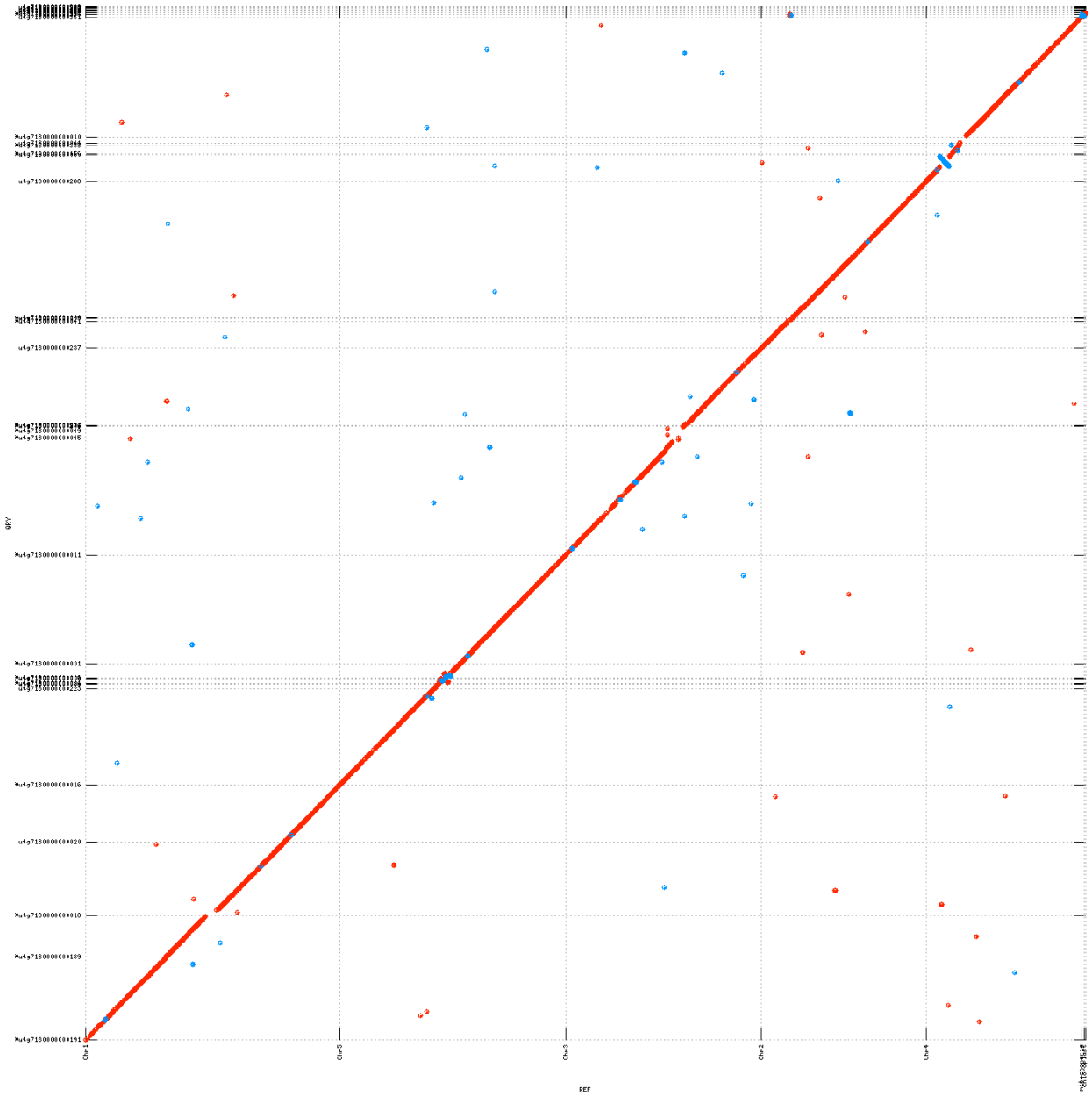

Figure S5. Mummerplot of the PBcR-MHAP assembly and *A. thaliana* reference
An alignment dotplot shows the relationship between the PBcR assembled contigs of *A. thaliana* Ler-0 (y-axis) and the *A. thaliana* Col-0 reference genome (x-axis). Contig and chromosome boundaries are displayed as dotted lines (horizontal and vertical, respectively), and contigs are ordered and oriented to match the reference using mummerplot's –fat option. The reference genome has been scaffolded, thus appearing more continuous in this plot. In terms of contigs, the PBcR assembly has better continuity than the reference assembly, with a larger N25, N75, and N95 contig size. A small, strain-specific inversion is visible towards the top-right of the plot, in blue.

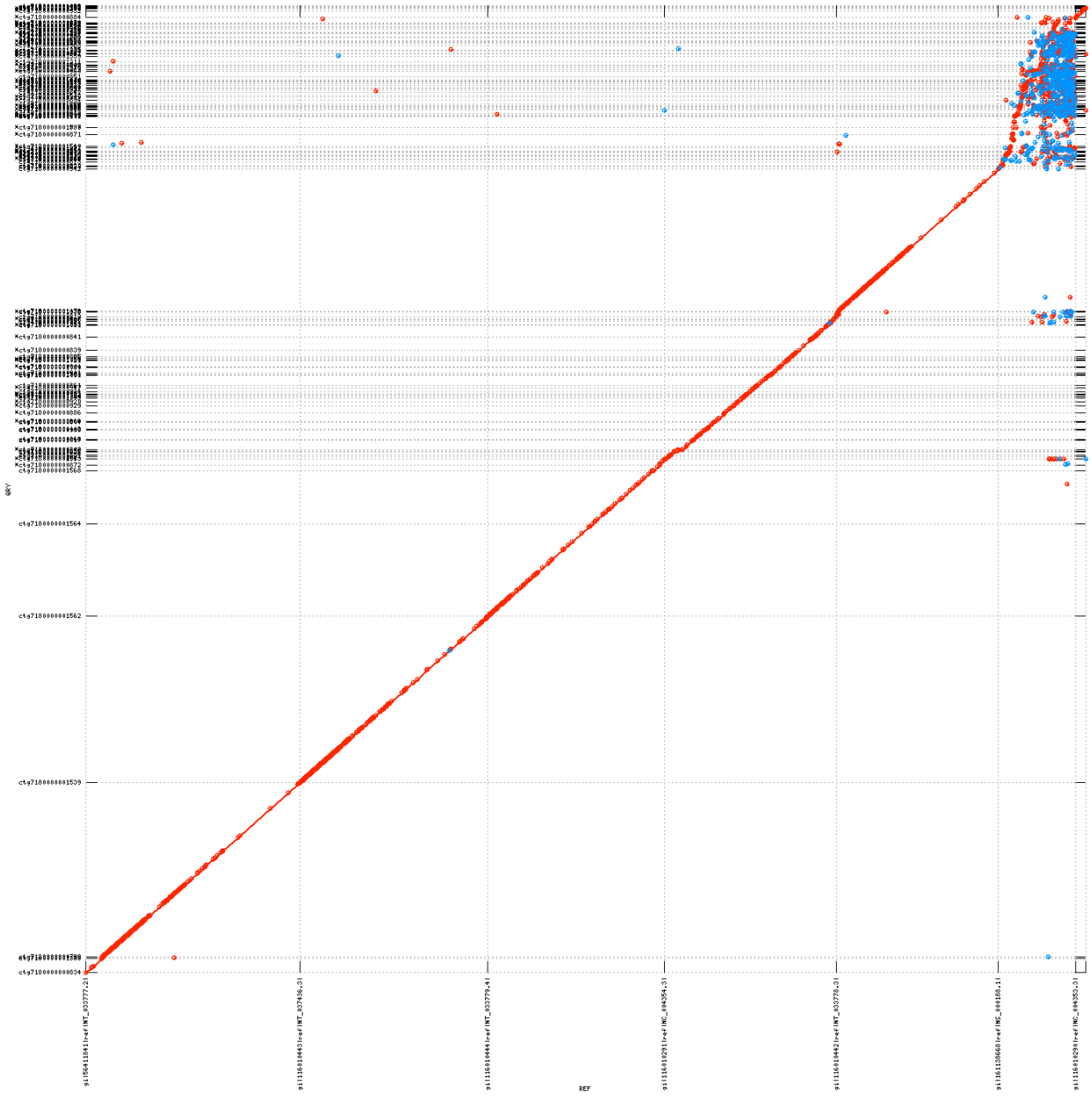

Figure S6. Mummerplot of the PBcR-MHAP assembly and *D. melanogaster* reference
An alignment dotplot shows the relationship between the PBcR assembled contigs of *D. melanogaster* ISO1 (y-axis) and the *D. melanogaster* ISO1 reference genome (x-axis). Contig and chromosome boundaries are displayed as dotted lines (horizontal and vertical, respectively), and contigs are ordered and oriented to match the reference using mummerplot's –fat option. Several PBcR contigs represent full chromosome arms, stretching into the telomeric and pericentromeric repeats. The more fragmented chromosome towards the middle of the plot is X, which was present in this sample at only half coverage (only male flies were sequenced). The heavily fragmented alignments at the top-right are "unplaced" reference sequences, which are largely repetitive and not associated with a particular chromosome.

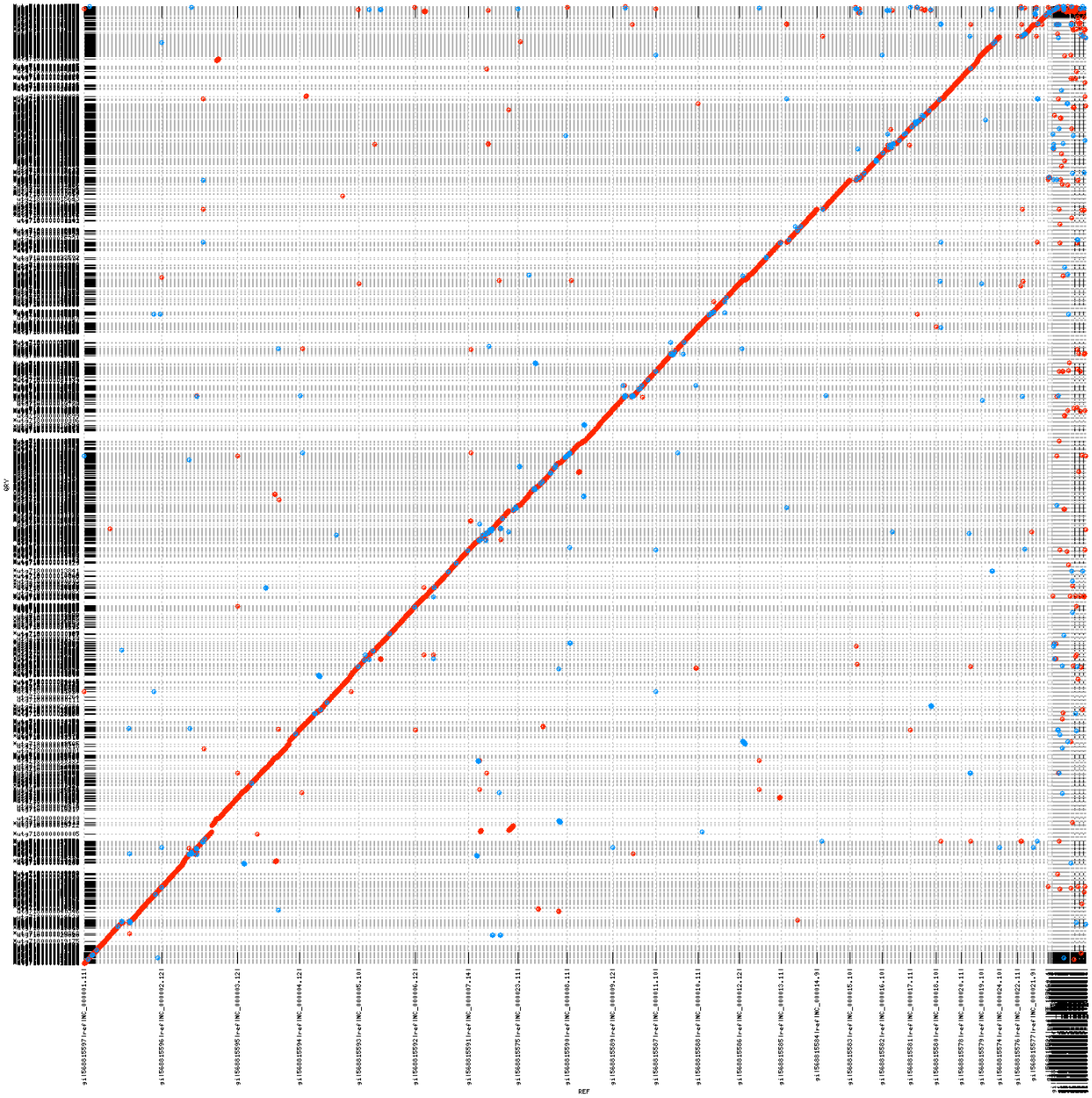

Figure S7. Mummerplot of the PBcR-MHAP assembly and human reference
An alignment dotplot shows the relationship between the PBcR assembled contigs of the CHM1htert cell line (y-axis) and the GRCh38 reference genome (x-axis). Contig and chromosome boundaries are displayed as dotted lines (horizontal and vertical, respectively), and contigs are ordered and oriented to match the reference using mummerplot's –layout option. The missing reference chromosome is Y.

Supplementary Note 5: Human assembly analysis

Genes falling within the MHC region (chr6:29655981- 33193468) were downloaded from UCSC [48] genome browser (http://genome.ucsc.edu/cgi-bin/hgTables?hgsid=382525021_VQMxM6OdyQG36vGeAVldr99Usmiu&boolshad.hgta_printCustomTrackHeaders=0&hgta_ctName=tb_refGene&hgta_ctDesc=table+browser+query+on+refGene&hgta_ctVis=pack&hgta_ctUrl=&fbQual=whole&fbUpBases=200&fbExonBases=0&fbIntronBases=0&fbDownBases=200&hgta_doGetBed=get+BED).
Each individual gene sequence was extracted from the reference and mapped with Nucmer to the assembly contigs covering this region in the assembly. Split mappings were included and the gene coverage was computed as the sum of mapped bases, while identity computed as the average mapped identity for all bases.

To generate the tiling figures, sequences were aligned as in Supplementary Note 5 using Nucmer and a custom script to convert output to the format expected by coloredChromosomes.pl (http://sourceforge.net/projects/cchrom/) [72]:

```
python makeMappings.py ref.1coords 10000 > ref1.tiling
perl convertToChr.pl human.chr.map ref1.tiling human.lanes human.chrPos >
ref1.cfg
perl coloredChromosomes.pl --chromosomeSpec ref1.cfg -o ref1.ps
ps2pdf ref1.ps
```

Because Nucmer was run using unique match anchors (to accelerate the alignment), some repetitive sequence remained unaligned. To avoid showing these regions as gaps in the tiling, the conversion script chains together consecutive alignments from the same contig if the alignment gap in the reference is less than 10,000bp. Thus, breaks in the resulting tiling occur whenever a contig switch occurs, or there is a >10,000bp gap between two alignments of the same contig. The entire process (alignment and figure generation) can be reproduced using the scripts available on the MHAP home page (assuming perl, python, and MUMmer are in your path) by running:

```
sh makeHuman.sh ref.fasta asm.fasta
```

Which will generate a figure named ref.pdf.

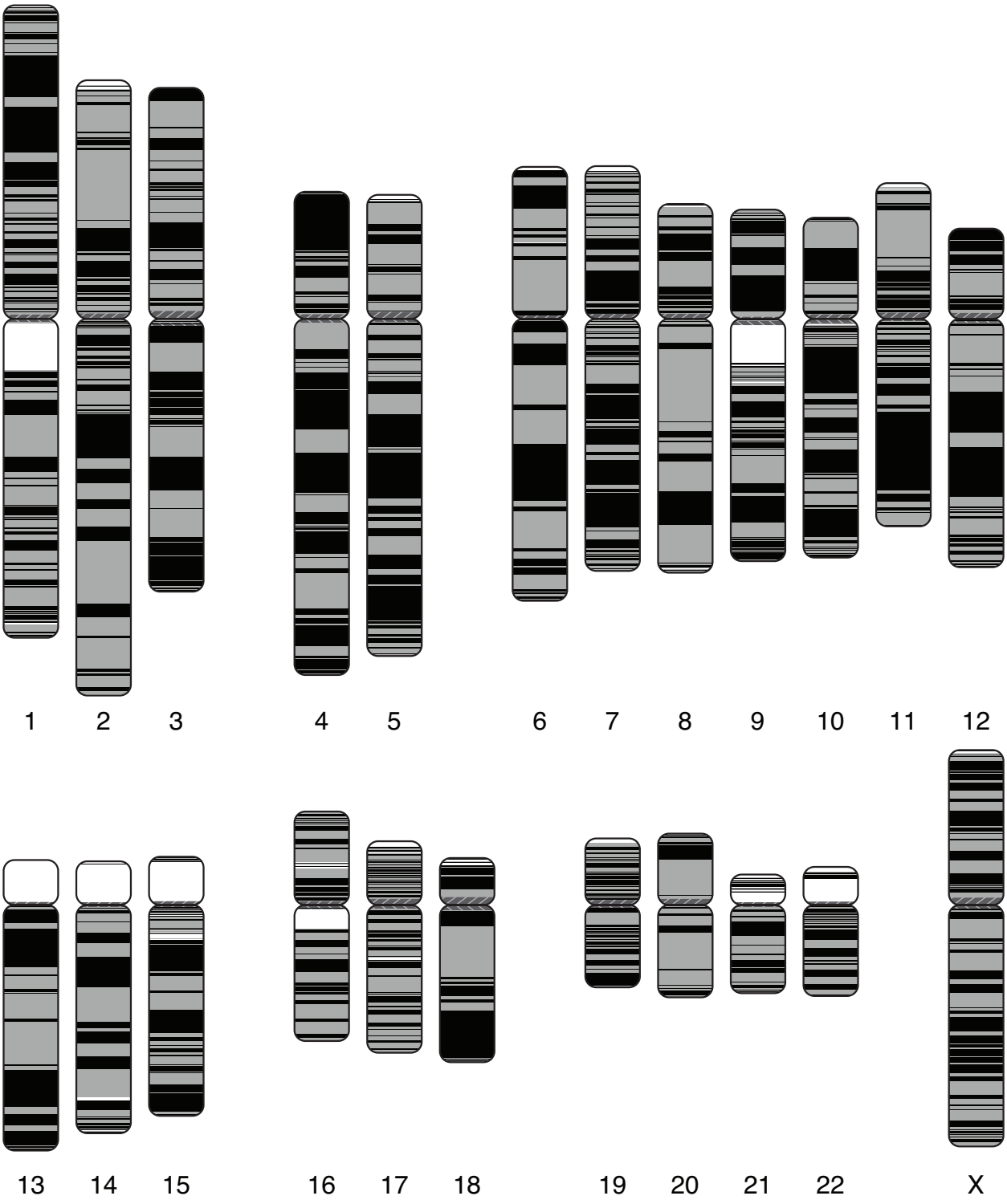

Figure S8. Contig tiling for MHAP-PBcR human CHM1 assembly
PBcR assembled contigs of the CHM1htert cell line are overlaid on human GRCh38 chromosomes. Alternating shades indicate adjacent contig mappings, so each transition from gray to black represents a contig boundary or alignment breakpoint. White indicates gaps in the tiling, the majority of which are uncharacterized reference sequence. Most chromosomes are covered by a few, large contigs that contiguously map to the reference.

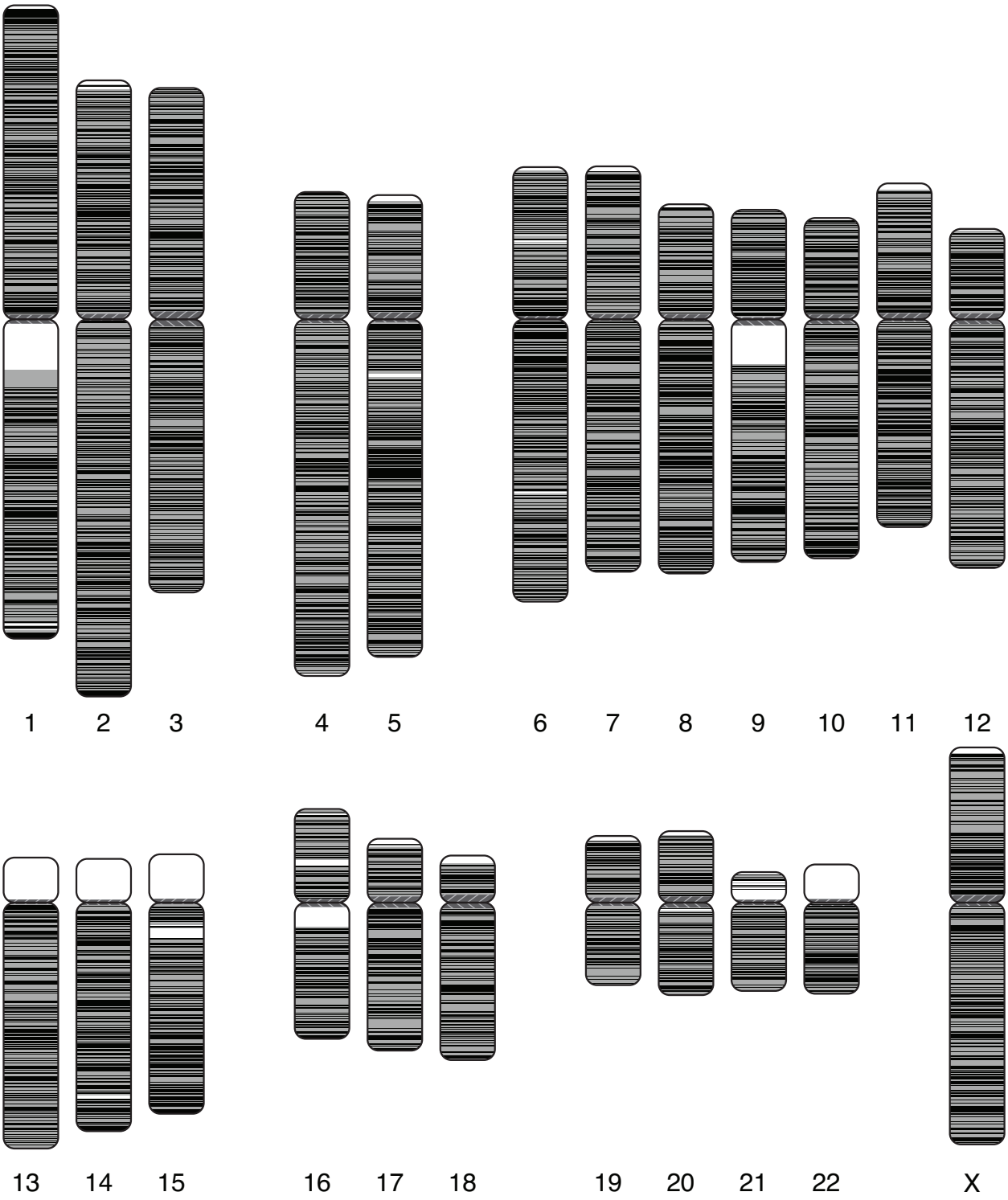

Figure S9. Contig tiling for the Illumina human CHM1 assembly
Assembled Illumina contigs of the CHM1htert cell line are overlaid on human GRCh38 chromosomes. Alternating shades indicate adjacent contig mappings, so each transition from gray to black represents a contig boundary or alignment breakpoint. White indicates gaps in the tiling, the majority of which are uncharacterized reference sequence. Most contigs are short, breaking each chromosome into thousands of contigs.

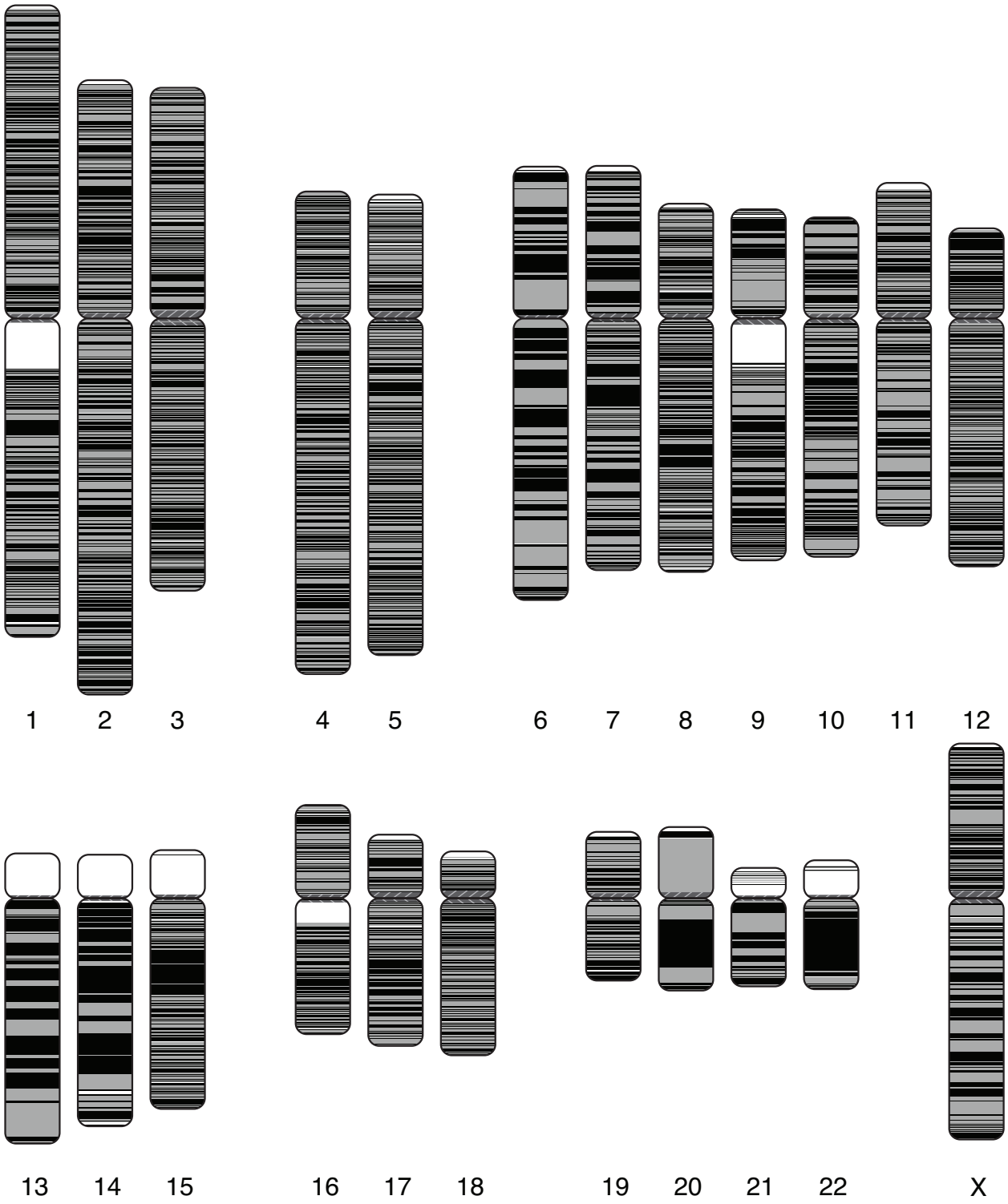

Figure S10. Contig tiling for the original hg1 human assembly
Contigs from the original Sanger assembly of human are overlaid on human GRCh38 chromosomes. Alternating shades indicate adjacent contig mappings, so each transition from gray to black represents a contig boundary or alignment breakpoint. There is a mix of long and short contigs, depending on the chromosome.

Supplementary Note 6: *D. melanogaster* assembly analysis

FlyBase 5.57_FB2014_03 was downloaded from ftp://ftp.flybase.net/genomes/dmel/dmel_r5.57_FB2014_03/fasta/dmel-all-gene-r5.57.fasta.gz. The gene sequences were aligned to the assembled contigs with the commands:

```
nucmer --maxmatch asm.all.fasta dmel-all-gene-r5.57.fasta
delta-filter -q out.delta > out.qdelta
show-coords -lrcTH out.qdelta > out.qcoords
```

High-quality alignments were defined as those >1Kbp in length and >99% identity. Euchromatic regions of the genome were defined as reference sequence assigned to 2L, 2R, 3L, 3R, 4, and X, while heterochromatic regions were defined as reference sequence assigned to 2LHet, 2RHet, 3LHet, 3RHet, XHet, YHet.

Genes contained within a single contig were counted as:

```
cat out.qcoords |awk '{if ($7 > 0 && $11 == 100) { print $NF} }' |wc -l
```

Genes contained within a single contig at >99 % identity:

```
cat out.qcoords |awk '{if ($7 > 99 && $11 == 100) { print $NF} }' |wc -l
```

Genes perfectly reconstructed:

```
cat out.qcoords |awk '{if ($7 >= 100 && $11 == 100) { print $NF} }' |wc -l
```

To analyze the TE repeat families, the analysis from [56]. The assembly features were downloaded from ftp://ftp.flybase.net/genomes/dmel/dmel_r5.57_FB2014_03/gff/dmel-all-r5.57.gff.gz. Assembled features matching "FlyBase transposable_element" were extracted and converted to bed, yielding 5,433 TE features. The assembled_feature_pipeline.sh was updated to speed up MUMmer computation by changing:

```
nucmer --maxmatch --prefix $WORKDIR/out $REFERENCE $ASSEMBLY
```

to:

```
nucmer -mumref -l 100 -c 1000 --prefix $WORKDIR/out $REFERENCE $ASSEMBLY
```

The pipeline was then executed as

```
assembled_feature_pipeline.sh -a asm.all.fasta -r reference.fasta -f dmel-all-trans.bed
```

The final output results/FINAL.REPORT was used to identify TEs with >= 100% Pct_length and corresponding Pct_ident. To identify repeat familes roo and juan, TEs with name matching roo{} and Juan, respectively were extracted from the final report.

Supplementary Note 7: Telomere assembly analysis

The *S. cerevisiae* S228 other_features database was downloaded from http://downloads.yeastgenome.org/sequence/S288C_reference/other_features/other_features_genomic.fasta.gz. The features were aligned to the assembly using the commands:

```
nucmer --maxmatch asm.all.fasta features.fasta
show-coords -lrcTH out.delta |sort -nk12 |awk '{if ($7> 85 && $11> 50) print $0}'|grep TEL |sort -rnk8 > tels.coords
```

Finally, contigs were identified that contain telomeric features within 50Kbp of the contig ends.

The *D. melanogaster* repeat libraries from [64] were downloaded and dmRepBase.fasta was used for all analysis. The repeats were mapped to the assembly with the commands:

```
nucmer --maxmatch asm.all.fasta dmRepBase.fasta
show-coords -lrcTH out.delta > out.coords
```

Any contigs with hits to *Het-A*, *TART*, and *Tahre* within 100Kbp of the contig ends were flagged as potentially containing telomeric sequences, 24 contigs passed the criteria. One contig contained *Het-A*, *TART*, and *Tahre* as well as *HetRp_DM*.

Supplementary Note 8: Quantifying overlap detection accuracy

Quantifying the accuracy of an overlapping method is important, since it provides a more direct method for understanding performance. However, rather than exhaustively computing the sensitivity of an overlapper by checking if all possible overlaps were detected, we randomly sampled the overlap space that exists when sequences are mapped to their best position in the reference. This measures the ability of an overlapper to determine if two reads originated from the same genomic locus, rather than an overlapper's ability to find "false" (repeat-induced) overlaps. In this case the estimated sensitivity can be expressed as the expected value, E[X], where *X* is the Bernoulli random variable with value 1 when the selected reference overlap is found in the overlapper's output, and 0 otherwise. We can estimate E[X] by randomly subsampling *X*, such that $E[X] \approx \bar{X} = \frac{1}{m}\sum_{i=1}^{m} X_i$ for a large enough *m*. The 95% confidence intervals [-δ,+δ] of the estimates $\bar{X}$ where derived using the Clopper-Pearson method [83]. To determine if the reference mapping of two reads overlapped, all sequences were mapped using BLASR:

```
blasr <sequences.fasta> <ref.fasta> -minReadLength 200 –nproc 32 –bestn 10 –m
4 –out <ref.m4>
```

To compute PPV, a similar sampling strategy is employed. A random overlap is selected from the program output and evaluated. If it is detected based on the reference matches, it is counted as a true positive. If it is not detected based on a reference matches, a full dynamic-programming Smith-Waterman alignment is perfomed on the overlapping segments and the overlap is is counted as a true positive if the identity is >70%. If *N* = number of samples, the PPV is then computed as the *TP/N*.

The sensitivity/specificity calculation is included in the MHAP distribution and can be run as:

```
java edu.umd.marbl.mhap.main.EstimateROC <mappingToReference.m4> <overlaps.m4
or overlaps.mhap> <fasta sequences> <min overlap> 10000 true false
```

Sensitivity and specificity on all datasets

Due to BLASR's long runtime, a maximum subset of 1Gbp (approximately 150K sequences) was randomly extracted from each of the five datasets. BLASR with k=12, bestn=1C, 10C, and 100C was run along with MHAP with default (k=16, h=512, min=3), and sensitive (k=16, h=1256, min=3). Sensitivity and specificity was evaluated as above. For BLASR the coverages correspond to 100X, 1,000X, 10,000X for *E. coli*; 85, 850, 8,500 for *S. cerevisiae*; 8, 80, 800 for *A. thaliana* and *D. melanogaster*; and 2, 20, 200 for human.

Supplementary Note 9: Consensus algorithm for correcting noisy, long reads

Corrected consensus sequences can be constructed by: (i) building a multiple sequence alignment (MSA) and (ii) deciding the correct base from the MSA columns [84, 89]. The errors in each individual read will have no or few support from other reads, while the correct bases will have many aligned bases from other reads. Encoding the MSA as partial order alignment [87] with a directed acyclic graph (DAG) is also an effective way for generating the consensus sequences. A consistent base-level alignment and the right partial order are essential to efficiently catch missing and extra bases correctly in the initial template sequence. A "tagging" and "sorting" approach described here uses the same principle to construct a consensus sequence. However, with this tagging and sorting approach, one does not need to construct a multiple sequence alignment or an alignment graph explicitly. The partial order relationship is maintained naturally in the process for generating the consensus sequence.

The inputs of the algorithm are (1) a template read **T** and (2) a set of reads denoted as $\mathbf{R_1}$ to $\mathbf{R_n}$ that are used to correct the template read. The output of the algorithm is a consensus sequence based on **T**. The sketch of this simple algorithm is as follows (Supplementary Figure S11):

(1) For each read $\mathbf{R_i}$, align it to the template **T** with an alignment algorithm developed by Myers [88]. Notice that there is no mismatch in the alignment output. Every alignment position is a single-base match, insertion, or deletion. Traditional mismatches become an insertion followed by a deletion.

(2) For each base **b** in $\mathbf{R_i}$, according to the alignment, we assign a tuple of number (*p, d*). If **b** is aligned as a match to *j*th base in **T**, *p* is assigned as *j* and *d* is assigned as 0. The tuple ((*p=j*, *d=0*), **b**) is inserted into a aligned tag list **B**. If there is no base aligned to *j*th base in T, a tuple ((*j*, 0), "-") is added into the aligned base list **B.** If **b** is an insertion relative to **T**, and the previous aligned base is on *j*th base in T, then $p = j$ and d = the number of bases from the previous aligned position. The tag ((*j*, $d \neq 0$), **b**) is added to **B**. See Figure S11 for an example.

(3) After all reads are aligned to T. We sort the elements in the list **B** by their numerical order for *p* and *d* followed by alphabetical order for **b**. To generate the consensus sequence, we count the number of for the same ((*p*, *d*), **b**). If the number is larger then half of the number of reads that are aligned across position *j* in T (= the number of element with *p=j*, *d*=0), we append the base **b** to the consensus sequence.

A heuristic is developed to handle templates with long stretches of errors or chimeric fragments formed by random ligation during sample preparation. In these cases, one might want to break the template at chimeric junctions or low quality regions to avoid propagating errors for downstream assembly steps. A sliding window counting the number of matches is used to decide if a region of the alignment in $\mathbf{R_i}$ should be used. A minimum

number of matches should be found with in the window. If the number is below a threshold then the alignment in the window will not be used. We scan the template sequence for each position to calculate the number reads can be use for correcting each region, if the number of reads is below a pre-specified threshold, we will break the template at those regions. Currently, the consensus sequence will only be generated for the longest contiguous region.

(a)

```
Inputs:

Reads,
read 1: ATATACGGC
read 2: ATCATCCGGC
read 3: ATATACCGAG
read 4: ATATAGCCGGC

Template:
T: ATATTAGGC
```

```
Alignments

read 1      ATAT-ACGGC
template    ATATTA-GGC
      p=    0123455678
      d=    0000001000

read 2      ATCAT--CCGGC
template    AT-ATTA--GGC
      p=    011234555678
      d=    001000012000

read 3      ATAT-ACCGAG-
template    ATATTA--G-GC
      p=    012345556678
      d=    000000120100

Read 4      ATAT-AGCCGGC
Template    ATATTAG--G-C
      p=    012345666778
      d=    000000012010
```

(b)

Sorted Tags	Count	Consensus Base
p,d,b		
0,0,A	4	A
1,0,T	4	T
1,1,C	1	-
2,0,A	4	A
3,0,T	4	T
4,0,-	4	-
5,0,A	3	A
5,0,-	1	-
5,1,C	3	C
5,2,C	2	-
6,0,G	4	G
6,1,A	1	-
6,1,C	1	-
6,2,C	1	-
7,0,G	4	G
7,1,G	1	-
8,0,C	4	C

Final Consensus = ATATACGGC

Figure S11. An example of the FalconSense algorithm. (a) Four reads and one template as the input of the algorithm shown. The assigned *p* and *d* are shown along with the alignments between the reads and the template. (b) The sorted tag list **B** and the counts for each tag are shown. In this case, if a tag count is greater than 2, a consensus base will be generated. The final consensus sequence, ATATACGGC, is just a simple concatenation of the non-empty bases.

Supplementary Note 10: Assembling corrected reads

All assemblies ran on a cluster composed of AMD 6136 2.4GHz CPUs. Overlap jobs were configured to use no more than 32GB of RAM, the maximum available per node. Total CPU hours were obtained using SGE's qacct utility, with the time computed as the sum of user and UTIME and STIME divided by 3,600. Data for *E. coli, S. cerevisiae,* and *D. melanogaster* were filtered with default parameters in SMRTportal to generate input. Due to an import bug in SMRTportal, *A. thaliana* data was filtered using length=500, quality=0.8. Human data was downloaded in fastq format post-filtering from the PacBio DevNet website. Unfortunately, the raw SMRT data for human has not yet been made available by Pacific Biosciences, so Quiver could not be run on this assembly.

All PBcR assemblies, except human, used the same Celera Assembler spec file, reproduced below:

```
merSize=16
mhap=-k 16 -num-hashes 512 -num-min-matches 3 -threshold 0.04 -subsequence-
size 100000

useGrid=1
scriptOnGrid=1

ovlMemory=32
ovlStoreMemory=32000
threads=32
ovlConcurrency=1
cnsConcurrency=8
merylThreads=32
merylMemory=32000
ovlRefBlockSize=20000
frgCorrThreads = 16
frgCorrBatchSize = 100000
ovlCorrBatchSize = 100000


sgeScript = -pe threads 1
sgeConsensus = -pe threads 8
sgeOverlap = -pe threads 15 -l mem=2GB
sgeCorrection = -pe threads 15 -l mem=2GB
sgeFragmentCorrection = -pe threads 16 -l mem=2GB
sgeOverlapCorrection = -pe threads 1 -l mem=16GB
```

The pipeline was run as:

```
PBcR -l <genomeName> -s pacbio.spec -pbCNS -fastq <fastq input sequences>
genomeSize=<approximate genome size> sgeName=<genome name> "sge=-A <genome
Name>
```

Where <genomeName> was set to ecoli, yeast, dmel, athal, and human, respectively and genome size was set to 4,650,000; 12,000,000; 130,000,000; 130,000,000; 3,000,000,000, respectively.

For human, we observed that the data was lower identity (average 82.98%) than other datasets (e.g. *D. melanogaster* average = 85.48%) and increasing MHAP sensitivity improved assembly performance. Due to this and the lower coverage, the spec file was modified to be (changes to the original spec file in bold):

```
merSize=14
mhap=-k 14 -num-hashes 1024 -num-min-matches 3 -threshold 0.04 -subsequence-
size 100000

useGrid=1
scriptOnGrid=1

ovlMemory=32
ovlStoreMemory=32000
threads=32
ovlConcurrency=1
cnsConcurrency=8
merylThreads=32
merylMemory=32000
ovlRefBlockSize=20000
frgCorrThreads = 16
frgCorrBatchSize = 100000
ovlCorrBatchSize = 100000

sgeScript = -pe threads 1
sgeConsensus = -pe threads 8
sgeOverlap = -pe threads 15 -l mem=2GB
sgeCorrection = -pe threads 15 -l mem=2GB
sgeFragmentCorrection = -pe threads 16 -l mem=2GB
sgeOverlapCorrection = -pe threads 1 -l mem=16GB

# relax overlap parameters
asmOvlErrorRate=0.10
asmUtgErrorRate=0.07
asmCgwErrorRate=0.10
asmCnsErrorRate=0.10
utgGraphErrorLimit=0.07
utgGraphErrorRate=3.25
utgMergeErrorLimit=0.0825
utgMergeErrorRate=5.25
asmOBT=0
```

and the pipeline run as:

```
PBcR -l <genomeName> -s pacbio.spec -fastq <fastq input sequences>
genomeSize=<approximate genome size> sgeName=<genome name> "sge=-A <genome
Name>
```